



UNIZG-FER 222464
Web Architecture, Protocols, and Services



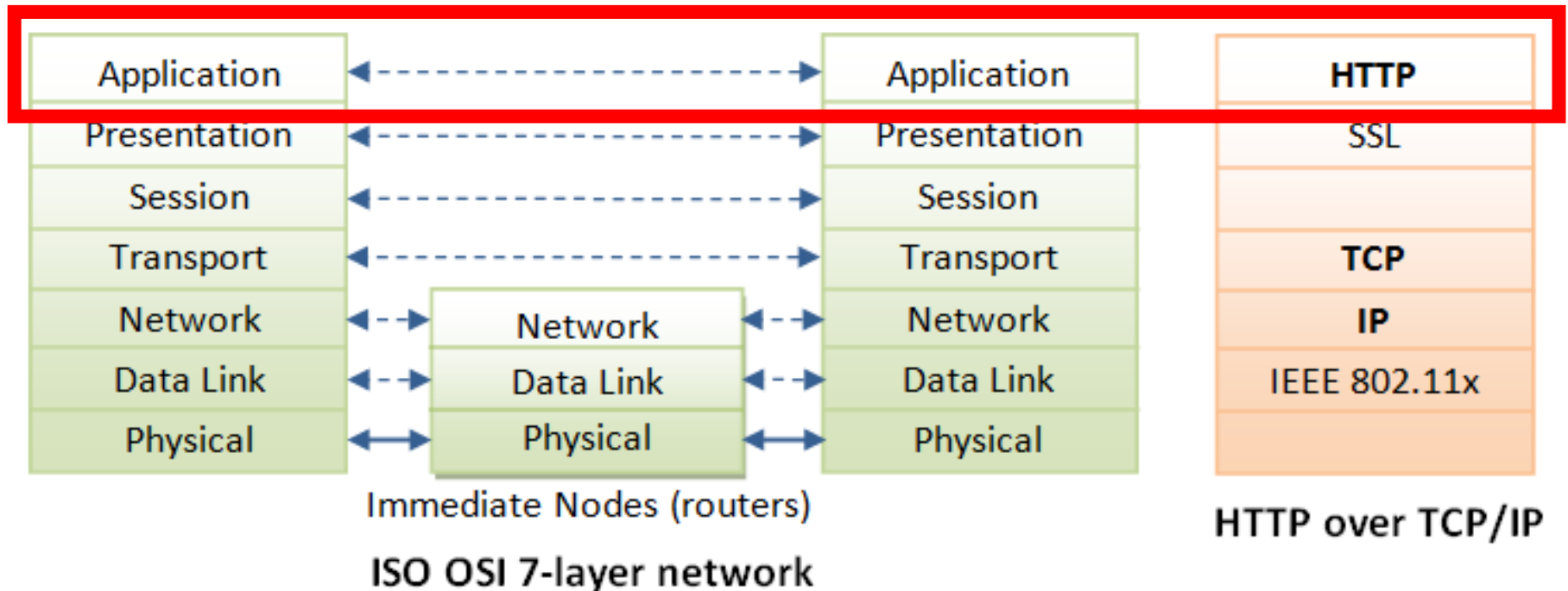
HTTP

The Driver of the World Wide Web

HTTP Basics



- HTTP (*HyperText Transfer Protocol*) protocol
 - Application-level protocol for distributed hypermedia information systems

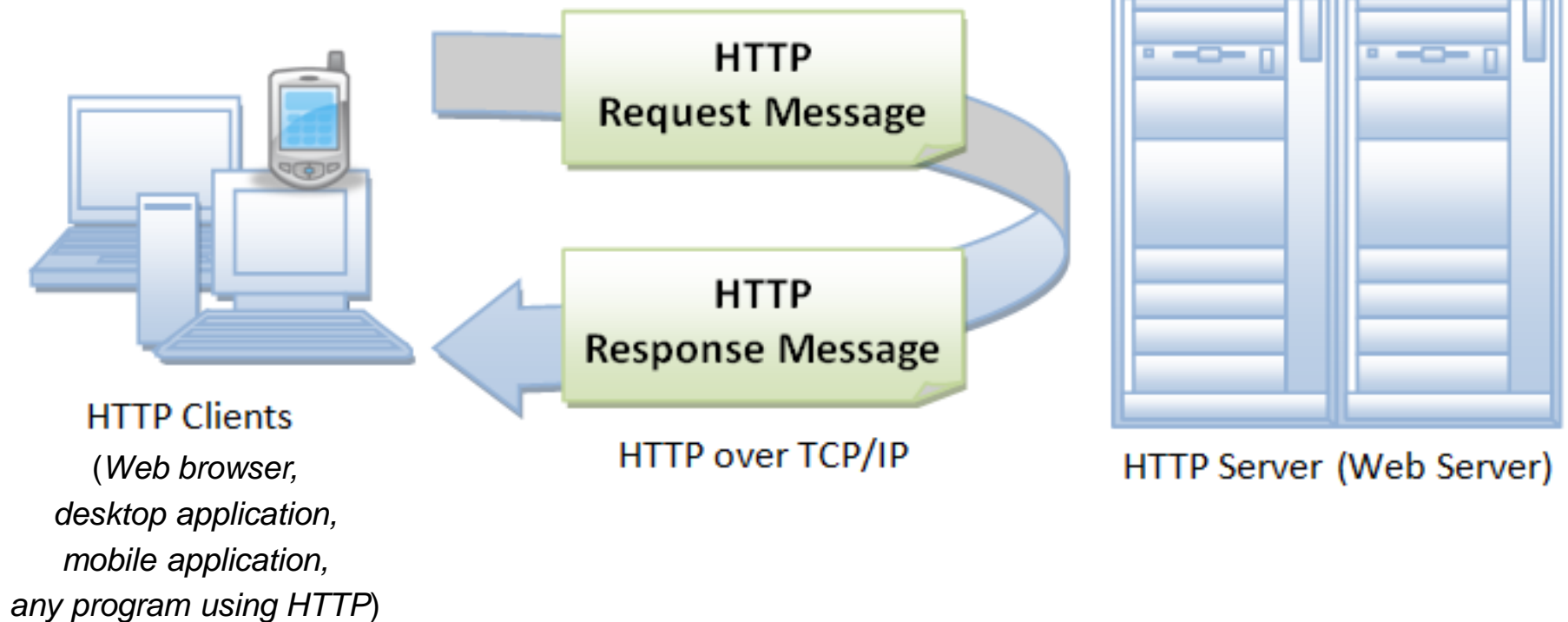


- **World Wide Web**
 - Applications and services based on HTTP protocol

HTTP Basics



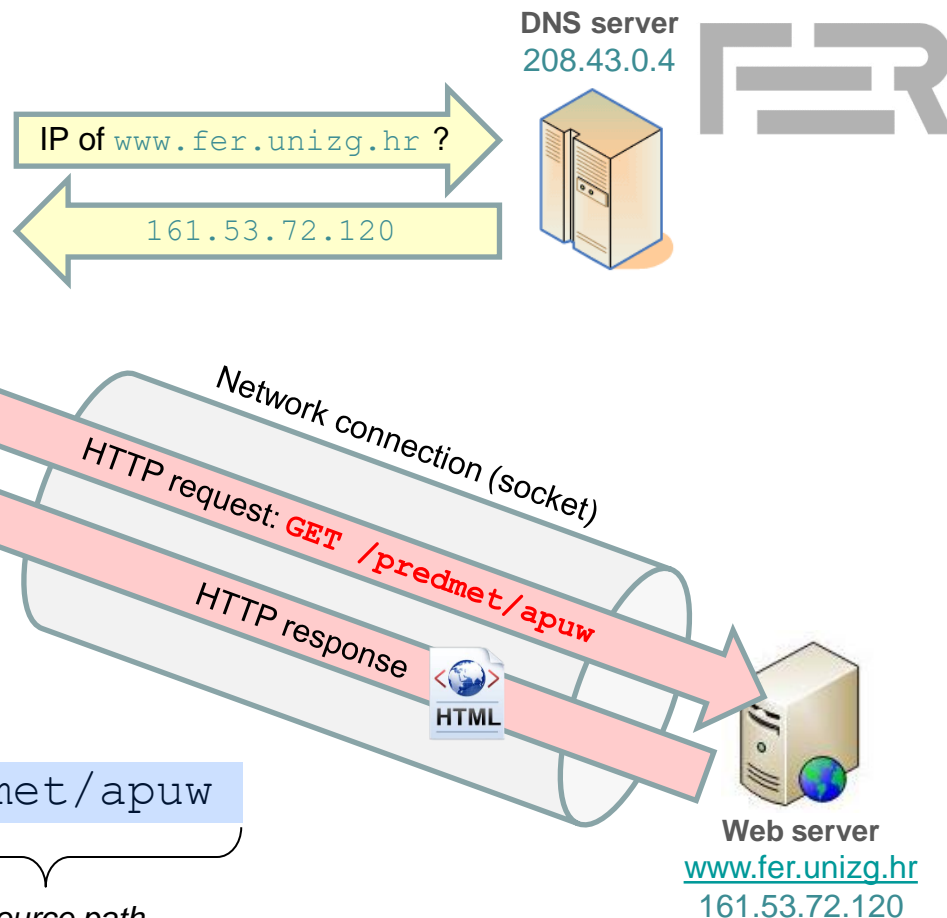
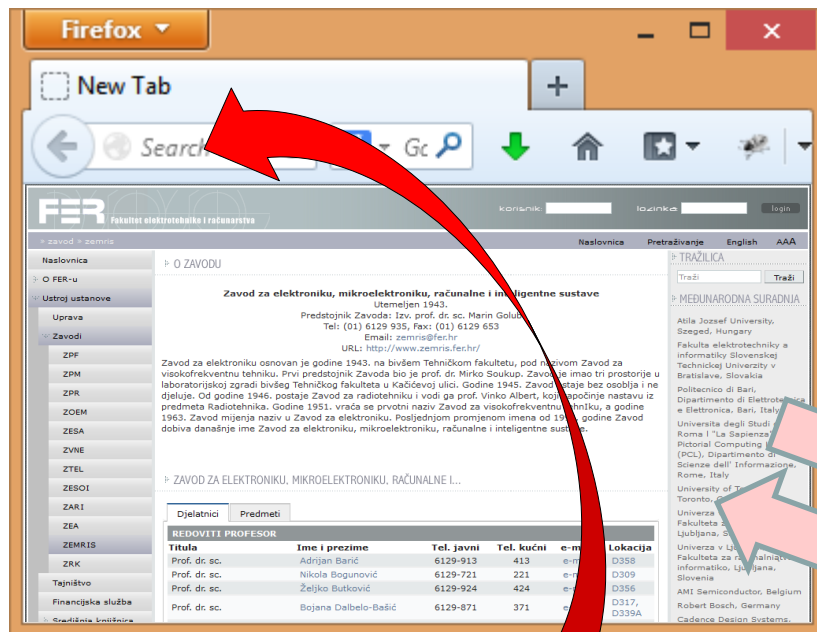
- HTTP (*HyperText Transfer Protocol*) protocol
 - Client-server architecture
 - Asymmetric request-response protocol
 - Client *pulls* information from the server
(instead of server *pushing* the information down to the client)



HTTP Basics



- Web browser
 - Most common HTTP client



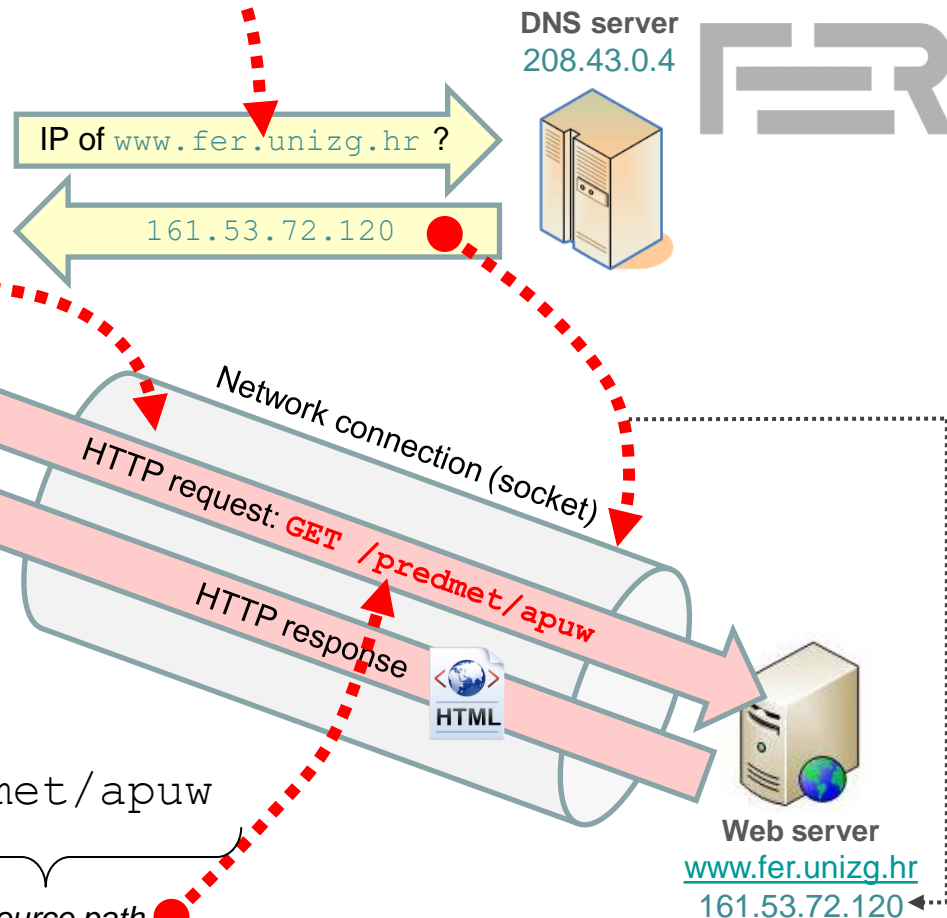
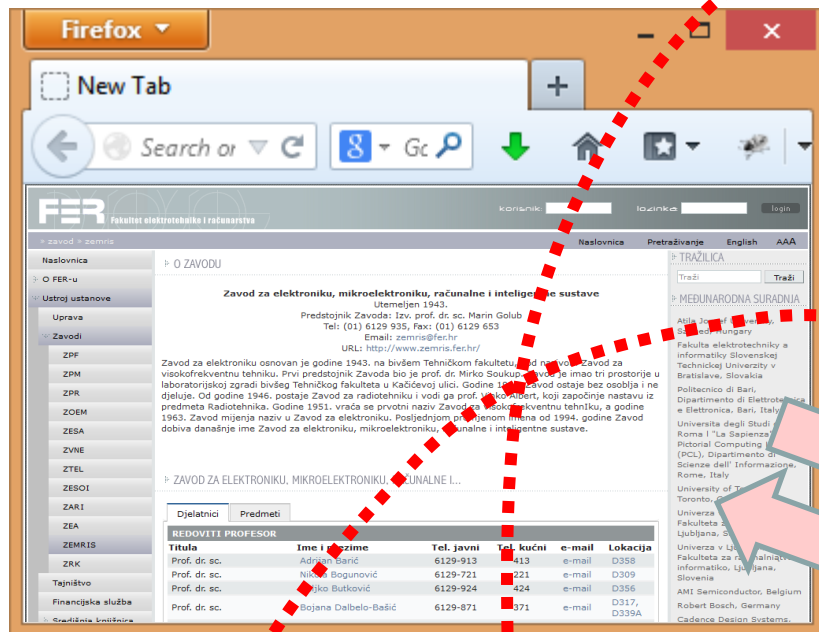
`http://www.fer.unizg.hr/predmet/apuw`

protocol *host* *resource path*

HTTP Basics



- Web browser
 - Most common HTTP client



`http://www.fer.unizg.hr/predmet/apuw`

protocol

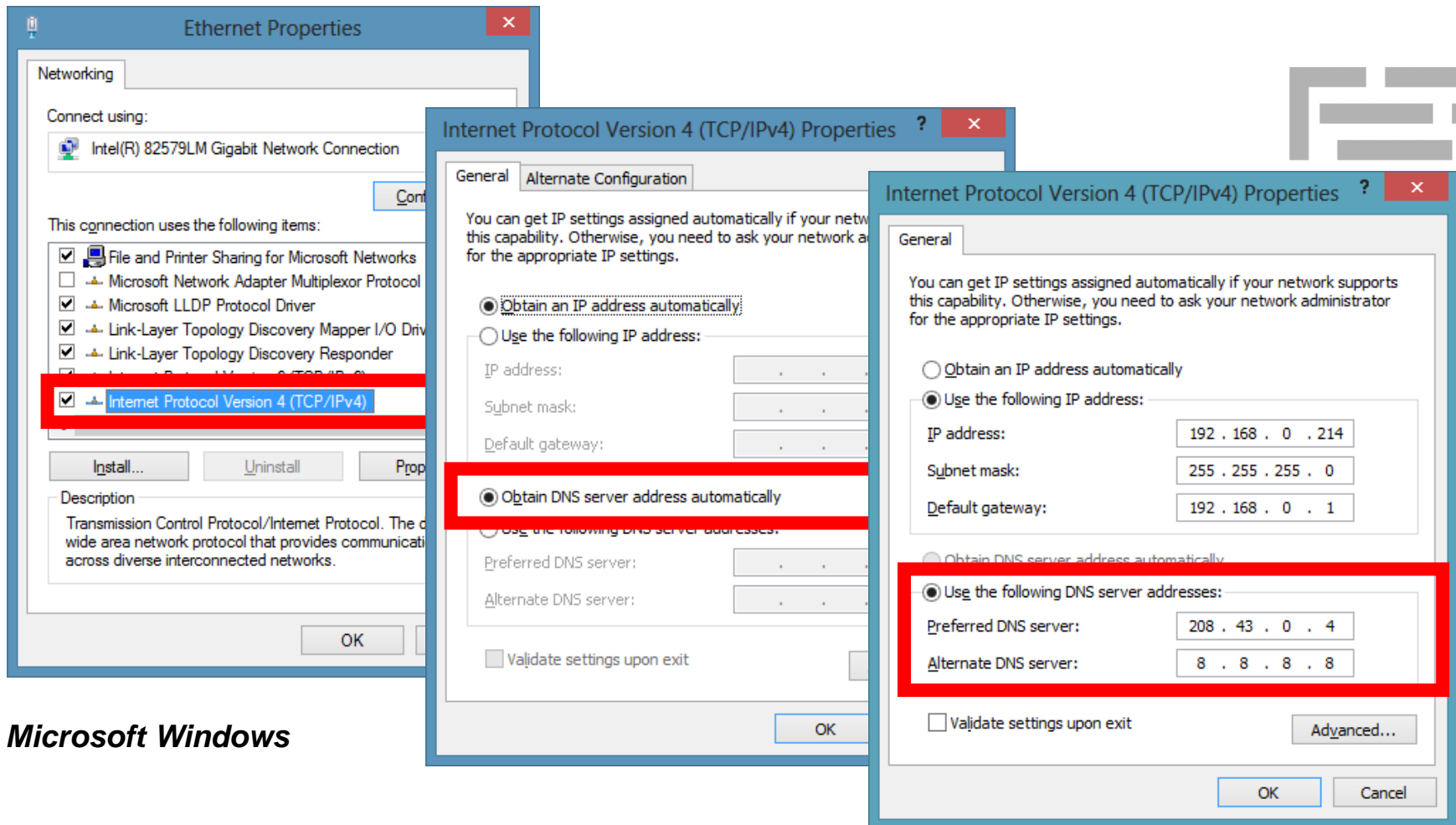
host

resource path

HTTP Basics



- How does a web browser find a DNS server?
 - Operating system network configuration

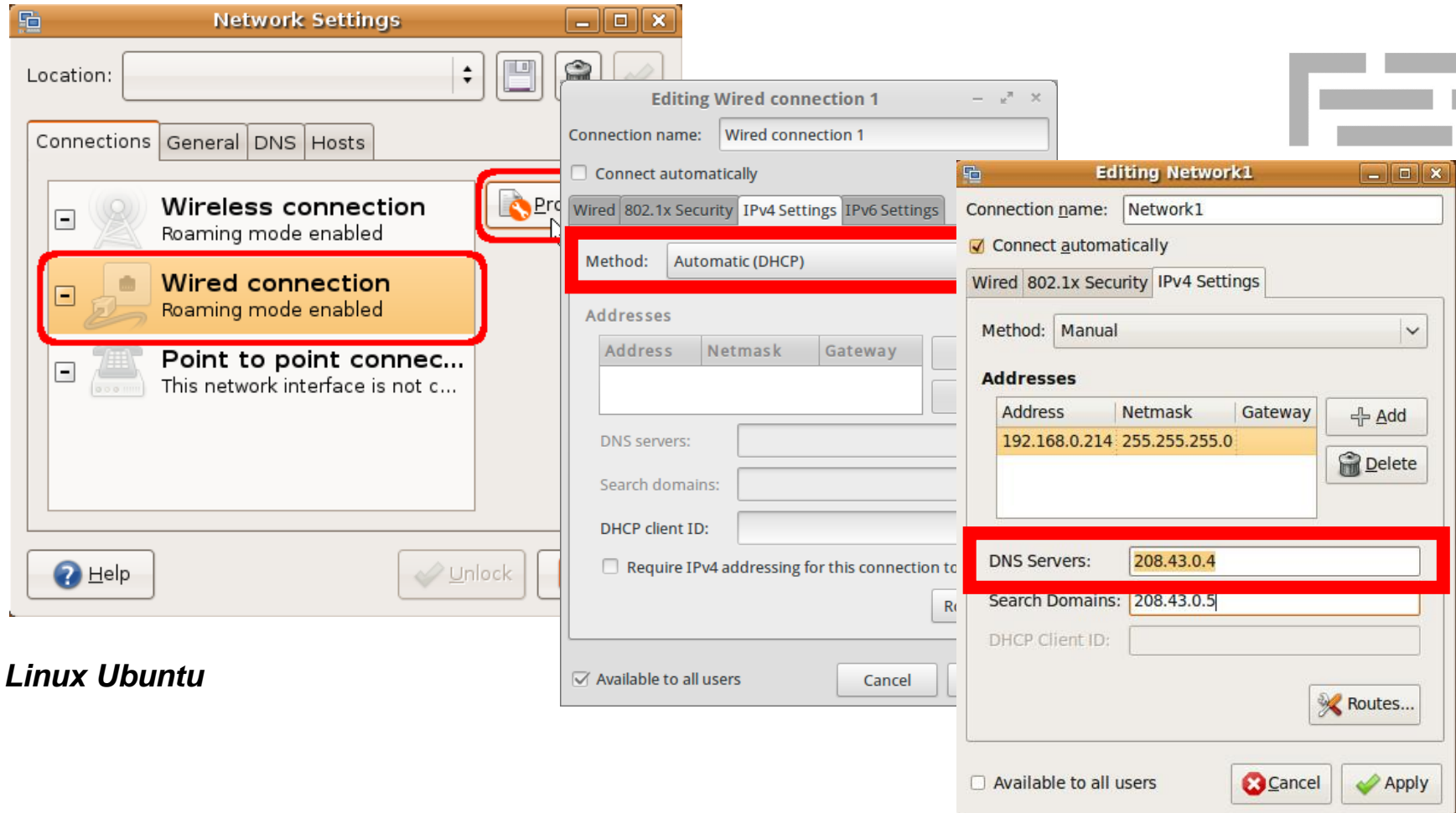


Microsoft Windows

HTTP Basics



- How does a web browser find a DNS server?
 - Operating system network configuration



Linux Ubuntu

HTTP Basics



- Uniform Resource Locator (URL)
 - String used to uniquely identify a resource on the web
- URL syntax

```
protocol://hostname:port/path-to-resource?parameters
```

protocol

- Application-level protocol used by the client and server
e.g. HTTP, HTTPS, FTP, telnet

hostname

- DNS domain name or IP address of the server
e.g. www.fer.unizg.hr, 161.53.72.120

port

- TCP port number the server is listening on for incoming requests from clients

path-to-resource

- Name and location of the requested resource under the server document base directory
e.g. static file on disk or program that dynamically renders the response

parameters

- Optional, used to additionally describe the resource (*we'll come back to this later*)

HTTP Basics



- URL examples

- 1) `http://www.fer.unizg.hr/predmet/apuw` (default HTTP port is 80)
`http://www.fer.unizg.hr:80/predmet/apuw`
`http://161.53.72.120/predmet/apuw`
`http://161.53.72.120:80/predmet/apuw`
- 2) `http://www.example.com:1234/europe/croatia/home.html`
- 3) `https://www.fer.unizg.hr/predmet/apuw` (default HTTPS port is 443)
`https://www.fer.unizg.hr:443/predmet/apuw`
- 4) `https://www.fer.unizg.hr:987/predmet/apuw`
- 5) `ftp://www.ftp.org/docs/test.txt` (default FTP port is 21)
- 6) `telnet://www.test101.com/` (default TELNET port is 23)



General HTTP client algorithm

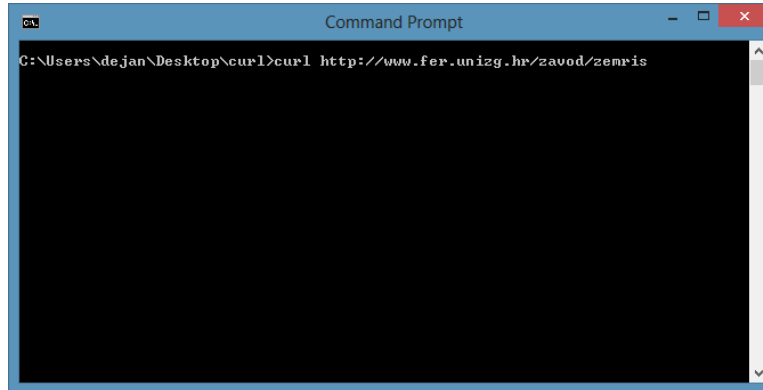
1. Client takes URL as input (*given by user or set programmatically*)
 2. Client parses the URL
 3. Client asks the DNS server for web server's IP address
 4. DNS server responds with IP address
- (steps 3 and 4 are not necessary if user enters web server's IP address instead of DNS name)*
5. Client opens a network connection to a given IP address and TCP port
 6. Client sends an HTTP request message to the web server
 7. Server maps the *resource path* part of the URL to a local file or program
 8. Server returns an HTTP response message
 9. Client processes the response (*e.g. web browser renders GUI and displays a web page to the user*)

HTTP Basics



- Other HTTP clients

- curl



- Command line syntax

`curl [options] <url>`

- Usage instructions

`curl -help`

- Simple HTTP request

`curl http://www.fer.unizg.hr/predmet/apuw`

`curl https://www.fer.unizg.hr/predmet/apuw`



HTTP Basics



- What happens on the network level?
 - Network monitoring & capture tool

The image shows a Wireshark 1.10.5 network packet capture window. The title bar indicates the capture is on the *Ethernet interface (Wireshark 1.10.5 (SVN Rev 54262 from /trunk-1.10)). The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, Internals, and Help. The toolbar contains various icons for file operations, capture control, and analysis. The filter bar shows 'http' as the selected filter. The packet list pane displays several packets, with packet 67 selected, showing an HTTP GET request to 161.53.72.120. The packet details pane shows the structure of the selected packet: Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol. The Hypertext Transfer Protocol section shows the GET request details, including the host (www.fer.unizg.hr), user-agent (Mozilla/5.0), and the full request URI (http://www.fer.unizg.hr/zavod/zemris). The packet bytes pane shows the raw data of the selected packet, with the first 28 bytes highlighted as a text item.

No.	Time	Source	Destination	Protocol	Length	Info
8	0.63422200	192.168.201.7	239.255.255.250	SSDP	139	M-SEARCH * HTTP/1.1
36	3.63448700	192.168.201.7	239.255.255.250	SSDP	139	M-SEARCH * HTTP/1.1
67	6.04841800	161.53.65.218	161.53.72.120	HTTP	364	GET /zavod/zemris HTTP/1.1
128	7.37321400	161.53.72.120	161.53.65.218	HTTP	571	HTTP/1.1 200 OK (text/html)
129	7.38755000	161.53.65.218	161.53.72.120	HTTP	451	GET /_pub1412605111/themes_static/metallish/
136	7.38824900	161.53.65.218	161.53.72.120	HTTP	457	GET /_pub1412605111/themes_static/metallish/
142	7.38848300	161.53.65.218	161.53.72.120	HTTP	393	GET /lib59/json2.js HTTP/1.1

Frame 67: 364 bytes on wire (2912 bits), 364 bytes captured (2912 bits) on interface 0

Ethernet II, Src: HewlettP_c3:58:7c (b4:b5:2f:c3:58:7c), Dst: Cisco_6a:97:40 (c8:4c:75:6a:97:40)

Internet Protocol Version 4, Src: 161.53.65.218 (161.53.65.218), Dst: 161.53.72.120 (161.53.72.120)

Transmission Control Protocol, Src Port: 64732 (64732), Dst Port: http (80), Seq: 1, Ack: 1, Len: 310

Hypertext Transfer Protocol

GET /zavod/zemris HTTP/1.1\r\n

Host: www.fer.unizg.hr\r\n

User-Agent: Mozilla/5.0 (windows NT 6.2; WOW64; rv:32.0) Gecko/20100101 Firefox/32.0\r\n

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n

Accept-Language: en-US,en;q=0.5\r\n

Accept-Encoding: gzip, deflate\r\n

DNT: 1\r\n

Connection: keep-alive\r\n

\r\n

[Full request URI: http://www.fer.unizg.hr/zavod/zemris]

0030 01 00 ce 0d 00 00 47 45 54 20 2f 7a 61 76 6f 64GE T /zavod

0040 2f 7a 65 6d 72 69 73 20 48 54 54 50 2f 31 2e 31 /zemris HTTP/1.1

0050 0d 0a 48 6f 73 74 3a 20 77 77 77 2e 66 65 72 2e ..Host: www.fer.

0060 75 6e 69 7a 67 2e 68 72 0d 0a 55 73 65 72 2d 41 unizg.hr ..User-A

0070 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e gent: Mo zilla/5.

0080 30 20 28 57 69 6e 64 6f 77 73 20 4e 54 20 36 2e 0 (windo ws NT 6.

0090 32 3b 20 57 4f 57 36 34 3b 20 72 76 3a 33 32 2e 2; WOW64 ; rv:32.

00a0 30 29 20 47 65 63 6b 6f 2f 32 30 31 30 30 31 30 0) Gecko /2010010

Text item (text), 28 bytes

Packets: 724 · Displayed: 113 (15.6%)

Profile: Default

HTTP Basics



- What happens on the network level?

HTTP request message

Web browser (Firefox)

```
GET /predmet/apuw HTTP/1.1
Host: www.fer.unizg.hr
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:23.0) Gecko/20100101 Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Cookie: __utma=161902635.17065694.1374136092.1377600687.1377615217.13; __utmz=...
Connection: keep-alive
```

curl

```
GET /predmet/apuw HTTP/1.1
User-Agent: curl/7.32.0
Host: www.fer.unizg.hr
Accept: */*
```

- What happens on the network level?

HTTP response message

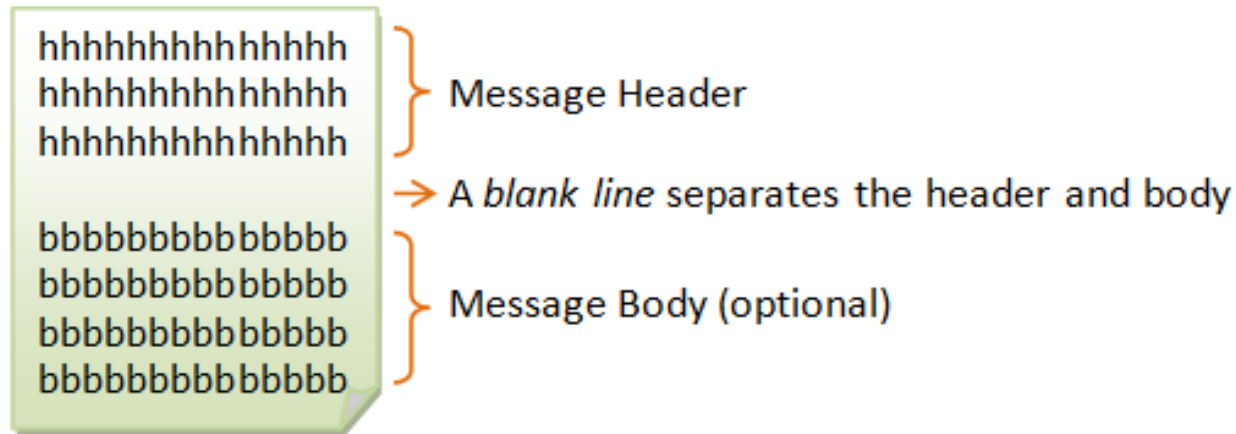
```
HTTP/1.1 200 OK
Date: Wed, 14 Oct 2021 10:49:28 GMT
Server: Apache/2.2.23 (FreeBSD) mod_fcgid/2.3.6 mod_ssl/2.2.23 OpenSSL/0.9.8x
X-Powered-By: PHP/5.3.19
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
P3P: CP="NOI CURa ADMa DEVa TAIa PSAa PSDa IVAa IVDa HISa OTPa ..."
Set-Cookie: CMS=vhtenqteedgso8d9u6l8h8vgq6; expires=Wed, 04-Sep-2013...
Content-Length: 73120
Content-Type: text/html; charset=utf-8

<!DOCTYPE HTML><html lang="hr" class="htmlcms"><head><meta http-equiv="Content-
Type" content="text/html; charset=utf-8" /><meta http-equiv="Content-Language"
content="hr" /><meta name="generator" content="Quilt CMS 2.5,
https://www.fer.unizg.hr/quilt-cms" /><!--meta name="robots" content="noindex"
/--><meta name="keywords" content="" /><title>Arhitektura, protokoli i usluge
weba</title><!-- breaks jquery-ui-1.10.3 tabs --><!--base
href="https://www.fer.unizg.hr/predmet/apuw"--><link rel="alternate"
type="application/rss+xml" title="FER: Arhitektura, protokoli i usluge weba"
href="/feed/rss.php?url=/predmet/apuw" ><link rel="alternate"
type="application/rss+xml" title="FER News: Arhitektura, protokoli i usluge
weba" href="/feed/rss.php?url=/predmet/apuw&portlet=news" >...
```

HTTP Basics



- HTTP messages
 - General form
 - Each HTTP message (either request or response) follows this general form

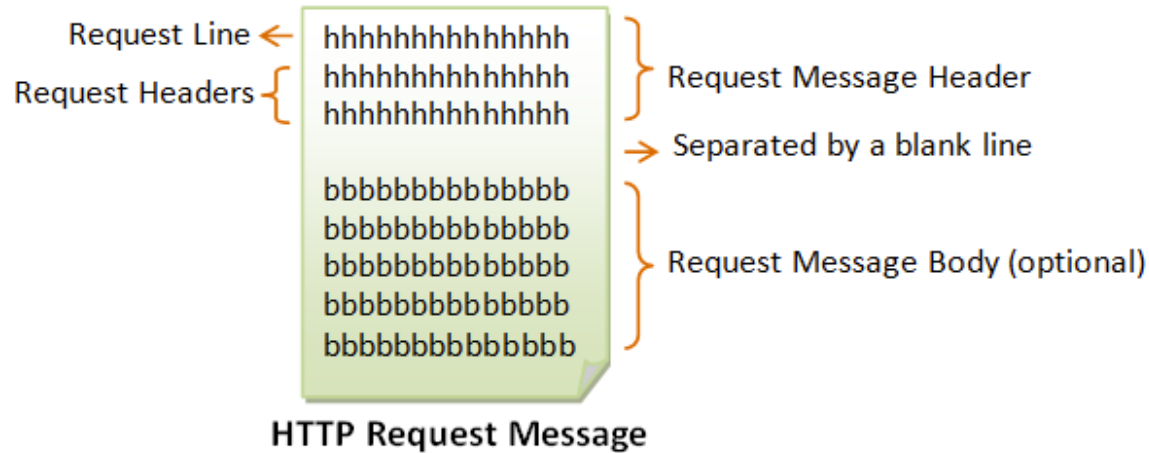


HTTP Messages

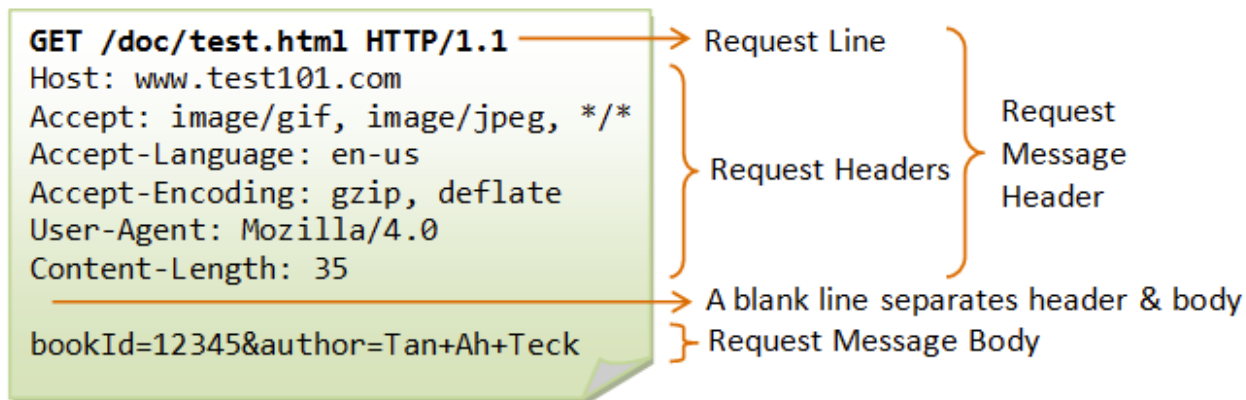
HTTP Basics



- HTTP request message



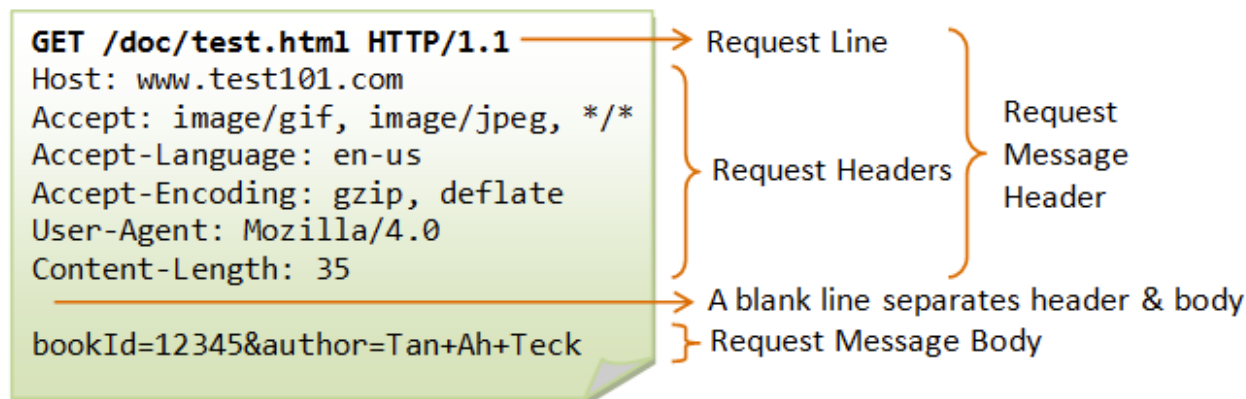
Example:



HTTP Basics



- HTTP request message



Request line:

`request-method-name request-URL HTTP-version CRLF`

request-method-name

Informs the server which operation to perform over the resource

HTTP protocol defines a set of request methods: GET, PUT, POST, DELETE, HEAD, and OPTIONS

The client uses one of these methods to send a request to the server

request-URL

Specifies the resource on a web server over which the server should perform the requested operation

HTTP-version

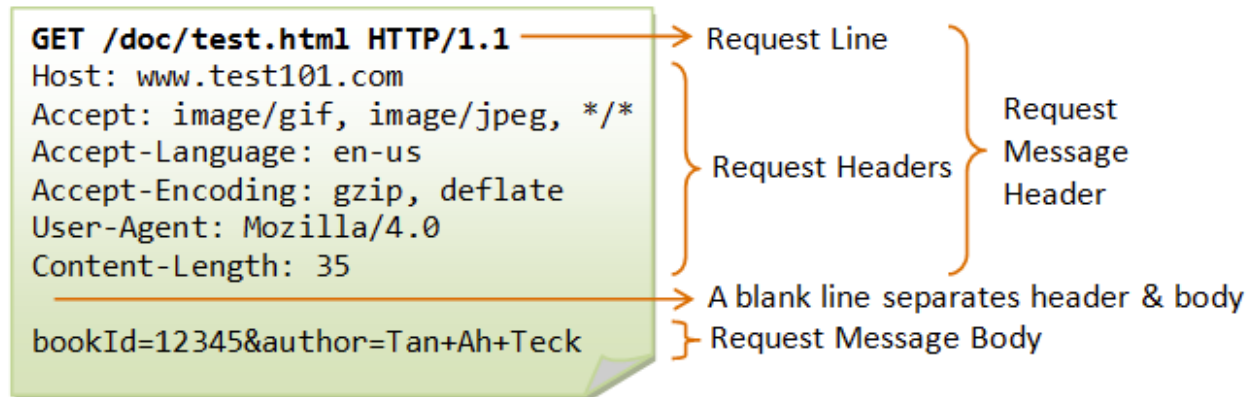
Client specifies the version of HTTP protocol it understands

Two versions are currently in use: HTTP/1.0 and HTTP/1.1 (recent HTTP/2 uses binary framing instead)

HTTP Basics



- HTTP request message



Request line:

request-method-name request-URL HTTP-version CRLF

Examples:

GET /predmet/apuw HTTP/1.1

GET /predmet/apuw HTTP/1.0

HEAD /predmet/apuw HTTP/1.1

PUT /photoalbum/image03.jpg HTTP/1.0

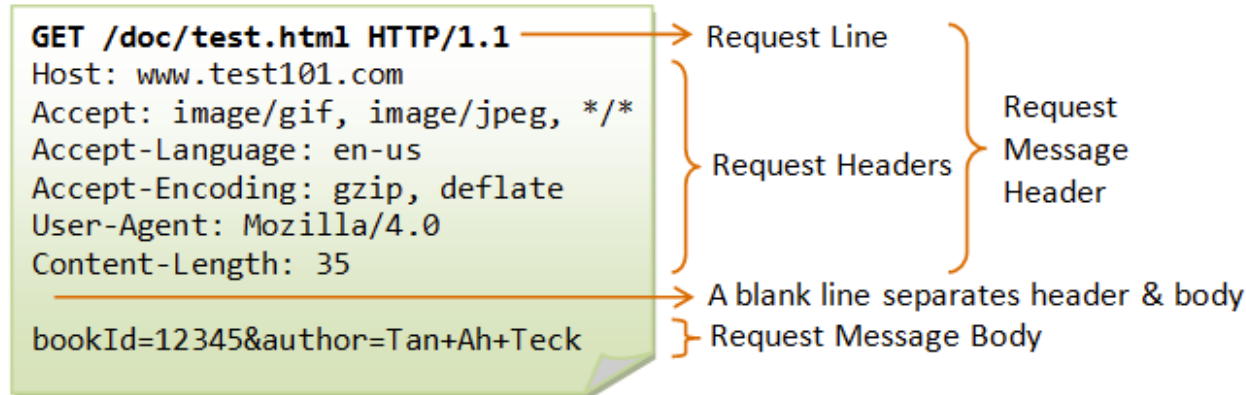
DELETE /photoalbum/image03.jpg HTTP/1.0

POST /news/article/comments HTTP/1.1

HTTP Basics



- HTTP request message



Request header:

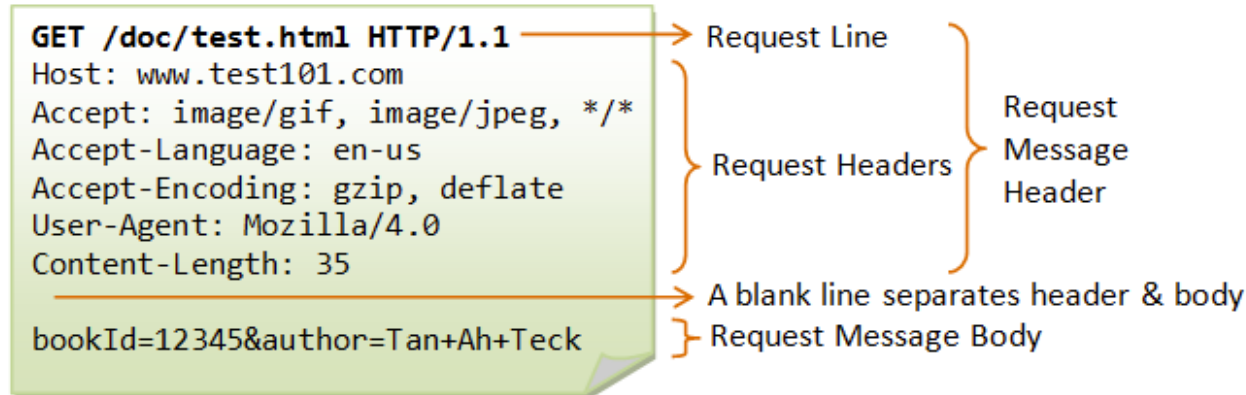
`request-header-name: request-header-value1, request-header-value2, ... CRLF`

- The request headers are in the form of `name: value` pairs
- Multiple header values, separated by commas, can be specified
- Each request header ends with a new line (CRLF)
- HTTP allows arbitrary number of request headers in a single request
- HTTP also allows custom non-standard header names
(*custom web servers might process custom headers, standard web servers ignore them*)

HTTP Basics



- HTTP request message



Request message body:

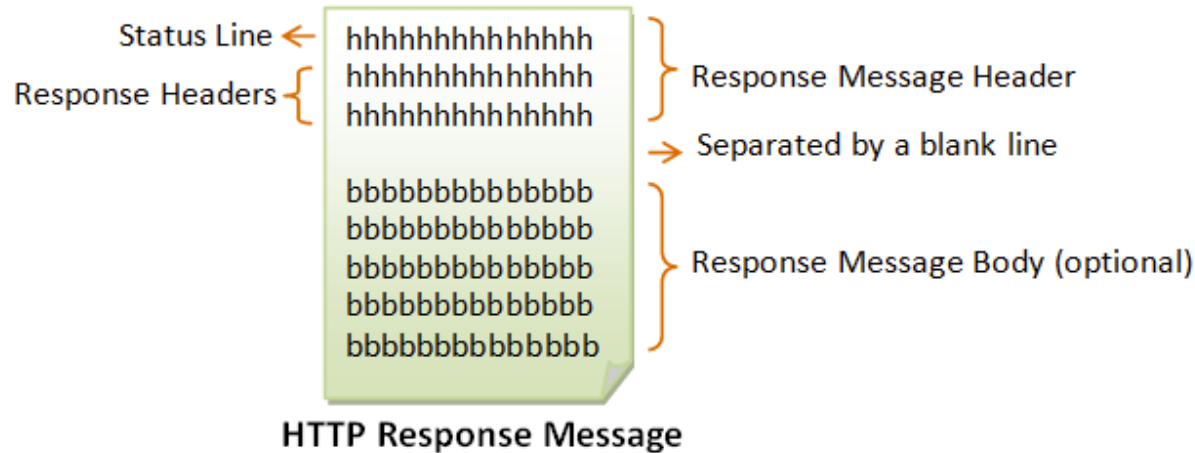
no defined structure, free format, arbitrary length

- Optional part of HTTP request message
- Used to send extra data with the request that cannot be specified in request headers (*for example, user-defined parameters*)
- HTTP protocol does not define the structure of request message body
- HTTP headers specify how to interpret the body (*for example, Content-Type header*)

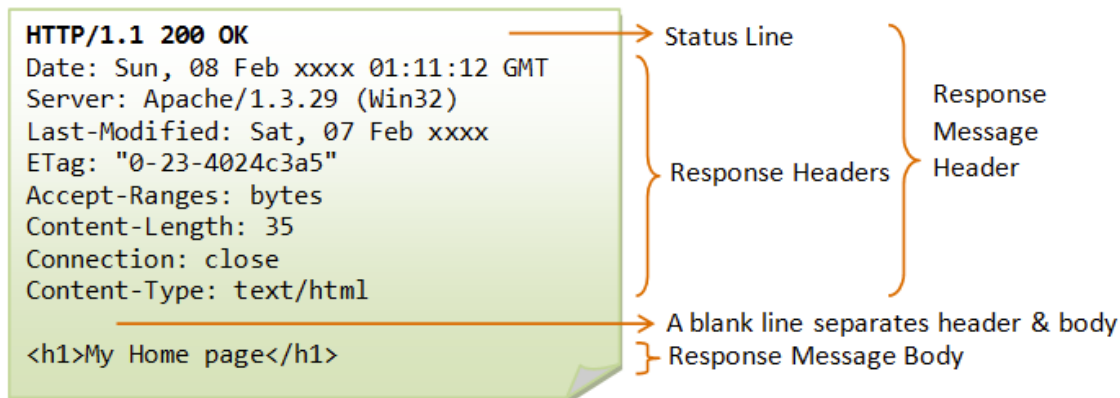
HTTP Basics



- HTTP response message



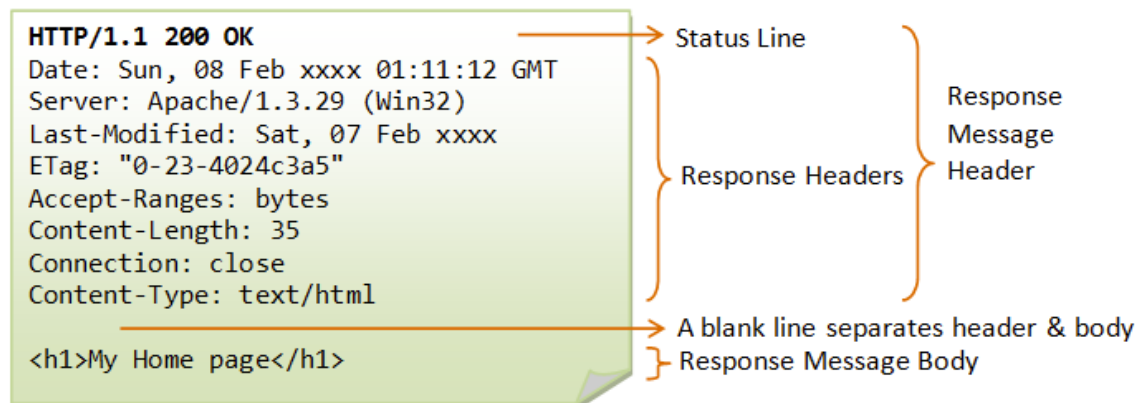
Example:



HTTP Basics



- HTTP response message



Status line:

HTTP-version status-code reason-phrase CRLF

HTTP-version

Server specifies the version of the HTTP protocol used in response

Version chosen by server should be equal or lower than the version specified in client's request

Two versions are currently in use: HTTP/1.0 and HTTP/1.1 (recent HTTP/2 uses binary framing instead)

status-code

A 3-digit number generated by the server to reflect the outcome of the request

Informs the client whether request is served successfully, some error occurred, etc.

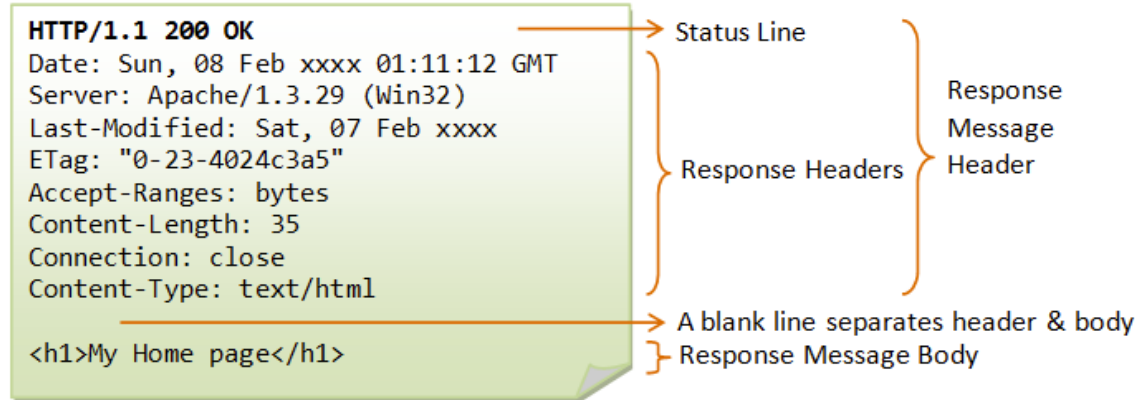
reason-phrase

Gives a short explanation to the status code

HTTP Basics



- HTTP response message



Status line:

HTTP-version status-code reason-phrase CRLF

Examples:

HTTP/1.1 200 OK

HTTP/1.0 404 Not Found

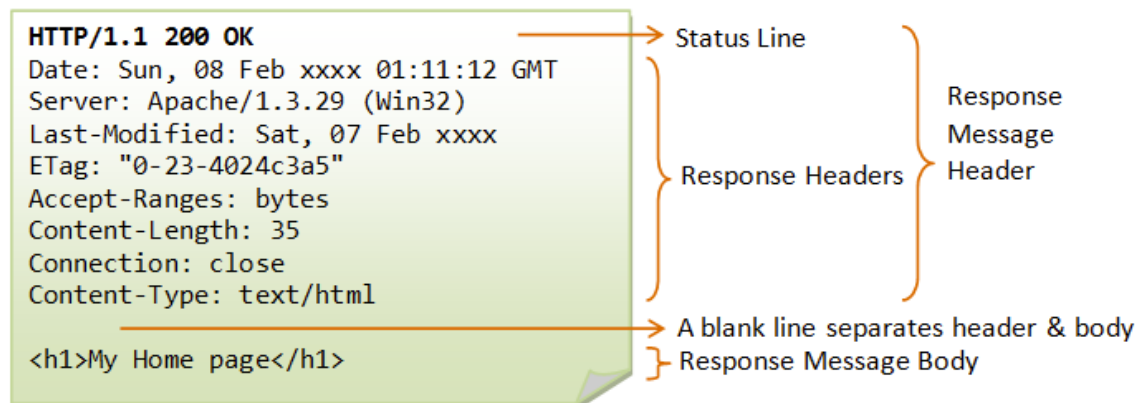
HTTP/1.1 403 Forbidden

HTTP/1.1 500 Internal Server Error

HTTP Basics



- HTTP response message



Response header:

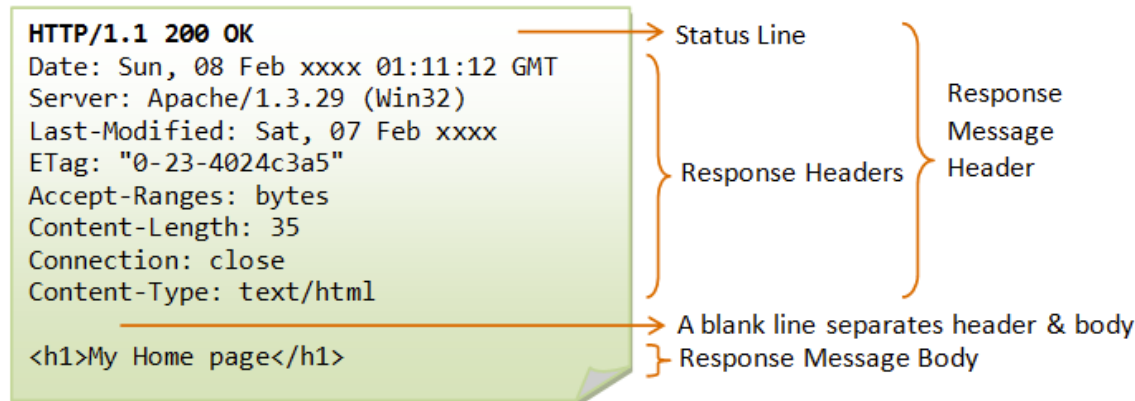
`response-header-name: resp-header-value1, resp-header-value2, ... CRLF`

- Response headers follow the same form as request headers
 - The response headers are in the form of `name: value` pairs
 - Multiple header values, separated by commas, can be specified
 - Each response header ends with a new line (CRLF)
 - HTTP allows arbitrary number of response headers in a single response
 - HTTP also allows custom non-standard header names
(*custom clients might process custom headers, standard HTTP clients ignore them*)

HTTP Basics



- HTTP response message



Response message body:

no defined structure, free format, arbitrary length

- Optional part of HTTP response message
- Used to send data from web server back to the client
(for example, web page's HTML, client-side script, image)
- HTTP protocol does not define the structure of response message body
- HTTP response headers specify how to interpret the body
(for example, Content-Type header)

- HTTP client socket-level programming

```
import java.net.*;
import java.io.*;

public class HttpClientSocket {
    public static void main(String[] args) throws IOException {
        // The host and port to be connected
        String host = "www.fer.unizg.hr";
        int port = 80;
        // Create a TCP socket and connect to the host:port
        Socket socket = new Socket(host, port);

        // Create the input and output streams for the network socket
        BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

        // Create request line
        out.println("GET /predmet/apuw HTTP/1.1");
        // Add some request headers
        out.println("Host: www.unizg.fer.hr");
        out.println("User-Agent: My custom HTTP client");
        // Add blank line separating header & body
        out.println();
        // Send request to the HTTP server
        out.flush();

        // Read the response and display on console
        String line;
        // readLine() returns null if server closes the network socket
        while((line = in.readLine()) != null) {
            System.out.println(line);
        }
        // Close the I/O streams
        in.close();
        out.close();
    }
}
```

HTTP Basics



- HTTP client socket-level programming

- Compile program

```
javac HttpClientSocket.java
```

- Start program

```
java HttpClientSocket
```





- HTTP client programming using HTTP library

```
import java.net.*;
import java.io.*;

public class HttpClientHttpLib {
    public static void main(String[] args) throws IOException {
        // The URL of the remote resource
        String url = "http://www.fer.unizg.hr/predmet/apuw";
        // Open a TCP connection for HTTP communication with a resource with given URL
        HttpURLConnection http = (HttpURLConnection) new URL(url).openConnection();

        // Read response status
        System.out.println("Response status code: " + http.getResponseCode());
        System.out.println("Response reason phrase: " + http.getResponseMessage());
        System.out.println();

        // Read response data if any
        String line;
        BufferedReader in = new BufferedReader(new InputStreamReader(http.getInputStream()));
        while((line = in.readLine()) != null) {
            System.out.println(line);
        }

        // Close the connection
        http.disconnect();
    }
}
```

HTTP Basics



- HTTP client programming using HTTP library

- Compile program

```
javac HttpClientHttpLib.java
```

- Start program

```
java HttpClientHttpLib
```

